

计算机学院学科基础课

---

# 计算机组成

## 时序电路

高小鹏

北京航空航天大学计算机学院  
系统结构研究所

# Great Idea #1: Levels of Representation/Interpretation

Higher-Level Language  
Program (e.g. C)

*Compiler*

Assembly Language  
Program (e.g. MIPS)

*Assembler*

Machine Language  
Program (MIPS)

*Machine  
Interpretation*

Hardware Architecture Description  
(e.g. block diagrams)

*Architecture  
Implementation*

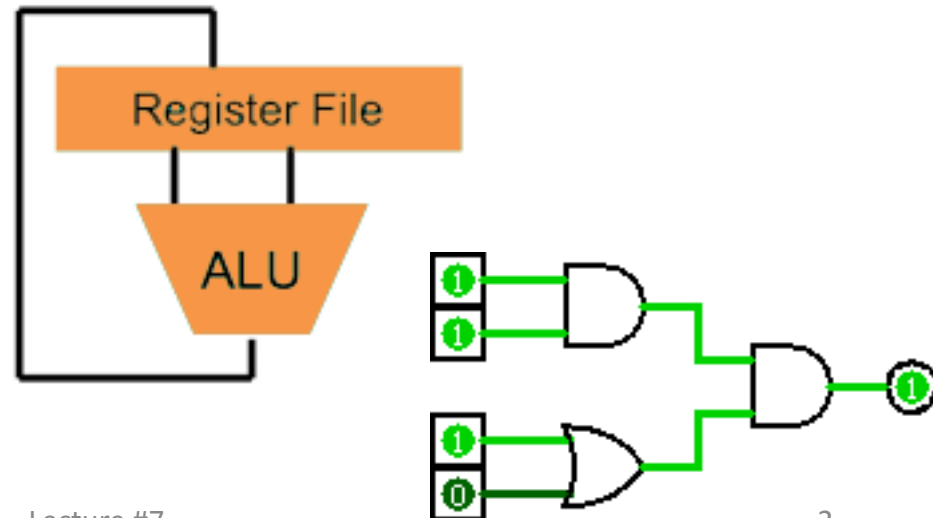
Logic Circuit Description  
(Circuit Schematic Diagrams)

```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

```
lw    $t0, 0($2)  
lw    $t1, 4($2)  
sw    $t1, 0($2)  
sw    $t0, 4($2)
```

We  
are  
here

```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```



# 提纲

- 内容主要取材

- CS617的18讲
- 《数字设计和计算机体系结构》的第2章
- 《数字设计和计算机体系结构》的第3章



# 组合逻辑电路的不足

## ❖ 组合逻辑电路的特点

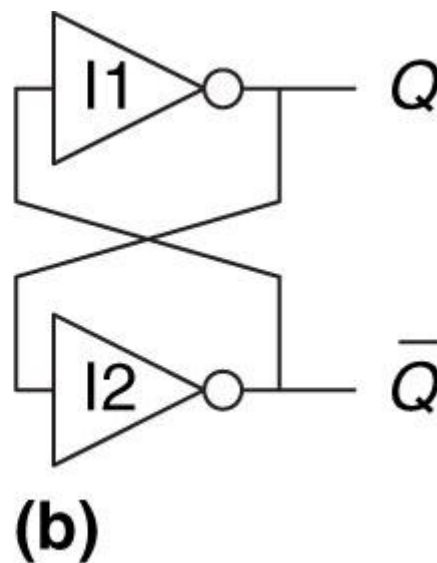
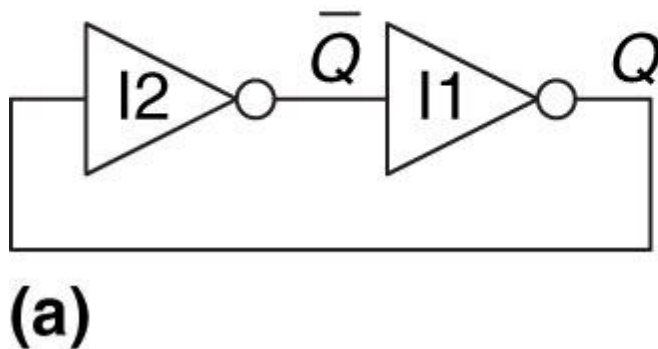
- ◆ 电路输出端的状态完全由输入端的状态决定
- ◆ 它是一种**无记忆**电路——输入信号消失，则输出信号也会立即消失

## ❖ 在数字系统中有时需要将参与（算术或逻辑）运算的数据和运算结果保存起来——在组合逻辑电路的输出端需要具有**记忆功能**的部件

## ❖ **触发器**就是构成记忆功能部件的基本单元，或者说是**实现存储（记忆）功能的基本单元电路**。

# 双稳态电路

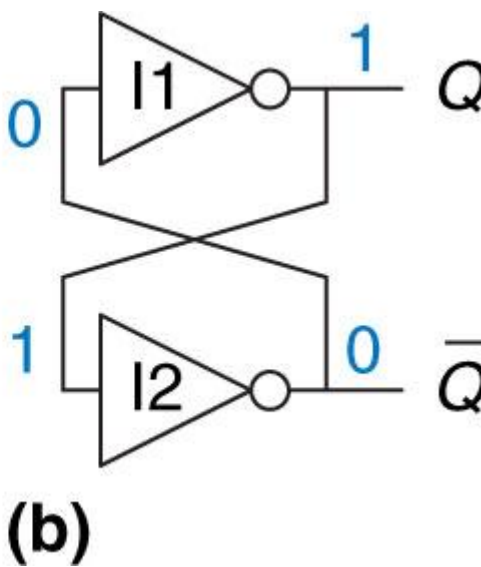
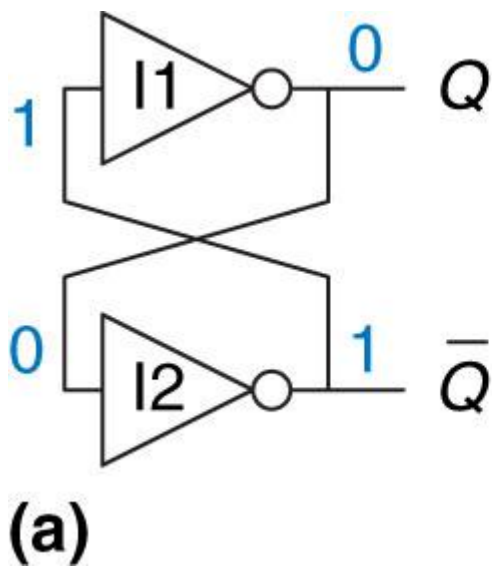
- 由2个反相器构成双稳态
  - 图b是图a的变形



# 双稳态电路

- 由2个反相器构成双稳态
  - 图b是图a的变形
  - 分析Q和/Q

Q	/Q	Q <sup>n</sup>
0	1	0
1	0	1

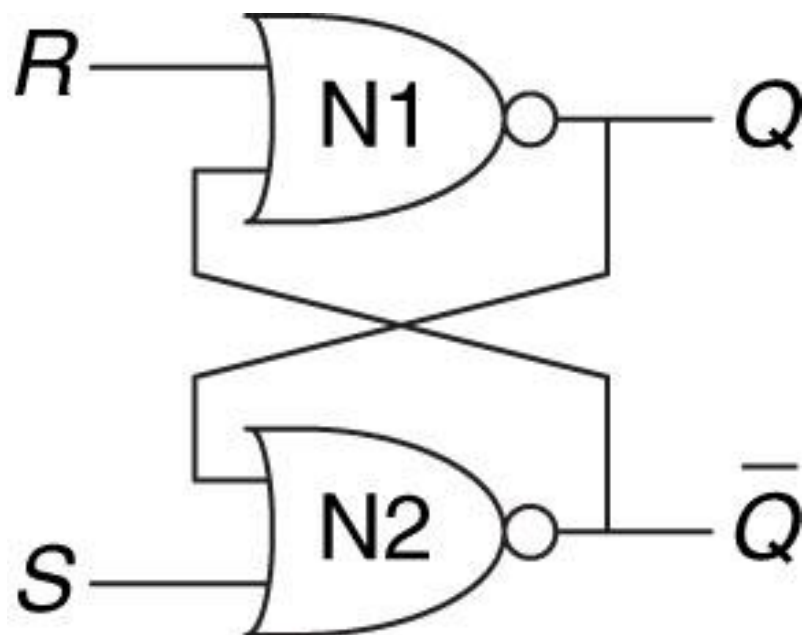


# 双稳态电路

- 双稳态电路
  - 可以保存1位信息
    - ◆  $Q=0$ ，则 $Q$ 永远为0
    - ◆  $Q=1$ ，则 $Q$ 永远为1
  - 如果 $Q$ 已知，则 $\neg Q$ 也已知
  - 首次加电，初值未知， $Q$ 未知
- 虽然可以保存1位信息，但没有实际价值
  - 没有输入，是一个Blackbox

# RS锁存器

- RS锁存器可以自行保持输出状态
  - ▣ 各种触发器的基本构成部分
- 分析RS锁存器的工作原理

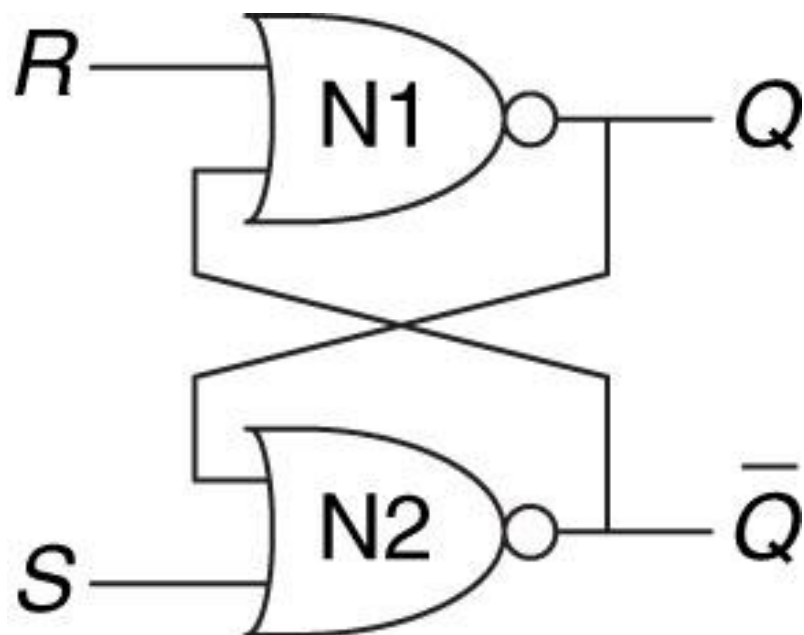


R	S	Q	/ $Q$	$Q^n$
1	0	0	1	0
0	1	1	0	1
1	1	0	0	0
0	0	?	?	?



# RS锁存器

- RS锁存器可以自行保持输出状态
  - ▣ 各种触发器的基本构成部分
- 分析RS锁存器的工作原理



S	R	Q	$\bar{Q}$	$Q^n$
0	1	0	1	0
1	0	1	0	1

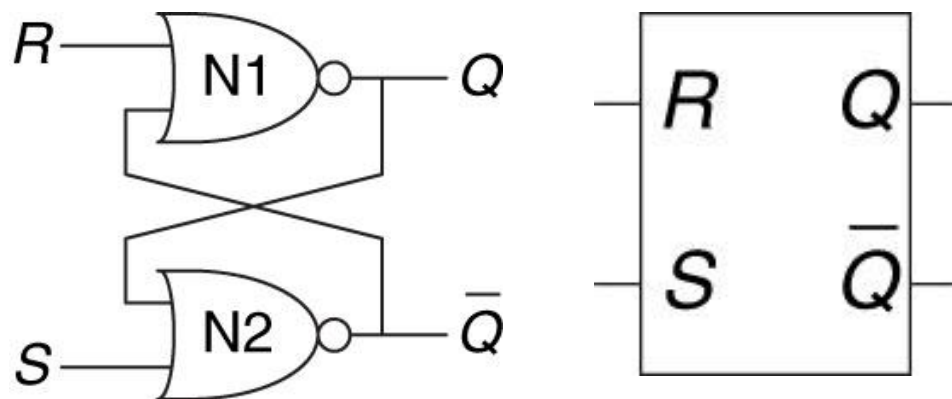
~~1~~   ~~1~~   ~~0~~   ~~0~~   ~~0~~

0	0	0	1	0
0	0	1	0	1

# RS锁存器

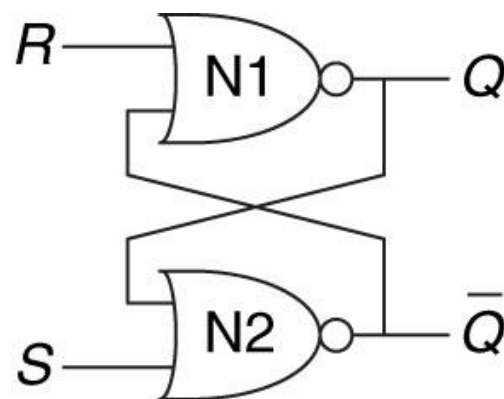
- 换一个符合记法
  - $Q_{prev}$  代表  $Q$ ,  $Q$  代表  $Q^n$
- 增加功能描述
  - 保持、清除、置位
- 注意：锁存器稳定工作状态不允许  $Q$  和  $\bar{Q}$  同值

	$S$	$R$	$Q$	$\bar{Q}$
保持	0	0	$Q_{prev}$	$\bar{Q}_{prev}$
清除	0	1	0	1
置位	1	0	1	0
非法	1	1	0	0



# RS锁存器

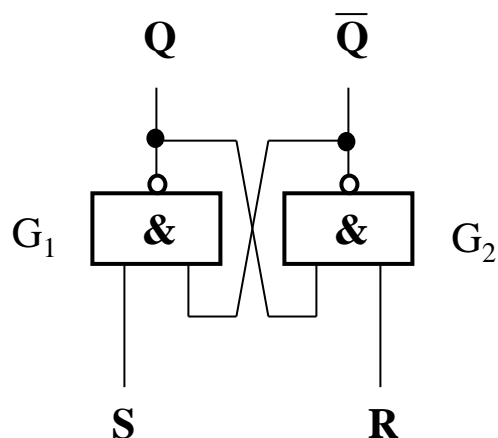
- 也可以将 $Q_{\text{prev}}$ 放在真值表输入部分
  - 锁存器输出也是输入的一部分
- $Q$ 和 $\bar{Q}$ ：更多时候只表达 $Q$
- 很多种表示方法
  - $Q$ 、 $Q^n$
  - $Q_{\text{prev}}$ 、 $Q$
  - $Q^n$ 、 $Q^{n+1}$
  - $Q$ 、 $Q_{\text{next}}$



功能	S	R	$Q_{\text{prev}}$	Q
保持	0	0		$Q_{\text{prev}}$
清除	0	1	0	0
置位	1	0	1	1
非法	1	1		

# RS锁存器

- 课堂练习：用与非门构造RS锁存器



功能	S	R	$Q_{\text{prev}}$	Q
保持	1	1		$Q_{\text{prev}}$
清除	1	0	0	0
置位	0	1	1	1
非法	0	0		

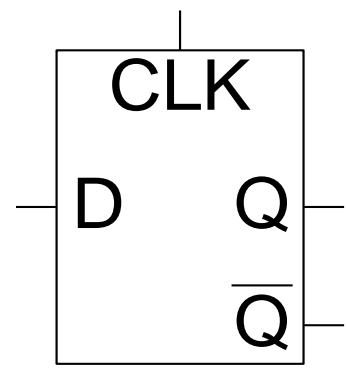
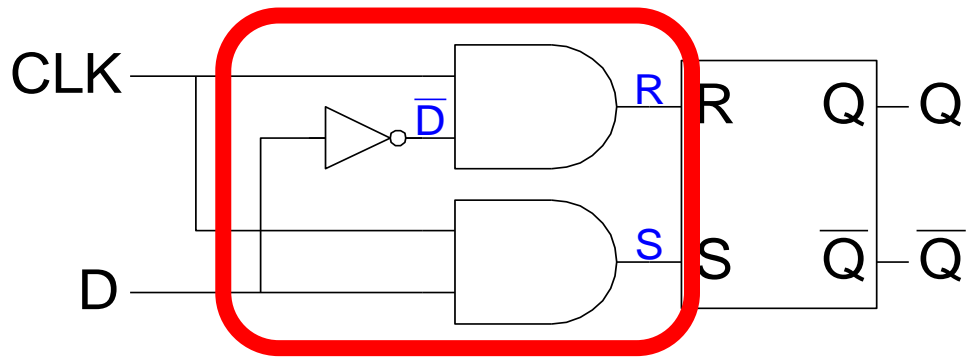
# RS锁存器的局限性

- 在数字系统中，为协调各部分电路运行，要求电路在**时钟信号**控制下统一动作
  - 即按一定节拍将输入信号反映在触发器的输出端
- RS锁存器本质：没有时间上的同步关系
  - 信号之间难以定序
  - R/S任一改变都可能造成Q改变
    - ◆ R/S的有效值、有效先后、时间长短均
- 结果：由于RS锁存器的值与时间无法分离，致使RS难以使用

# RS锁存器的局限性

- 解决思路：RS锁存器增加一个控制端
  - 只有在控制端有效时，RS锁存器才能动作
  - 即只有控制端有效时，RS锁存器输出由输入决定
- 这种控制信号通常为周期振荡信号：方波
  - 也称为时钟信号
  - CLK、Clock

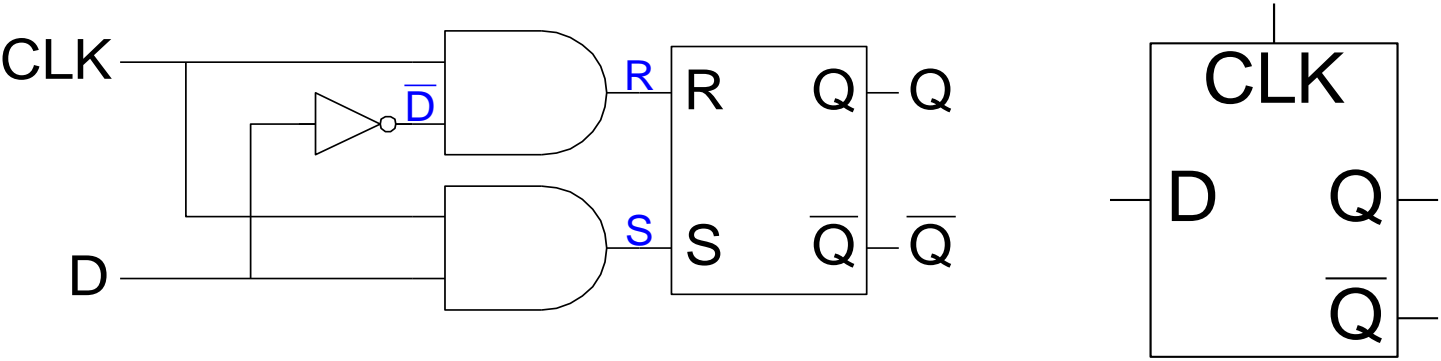
# D Latch Internal Circuit



$CLK$	$D$	$\overline{D}$	$S$	$R$	$Q$	$\overline{Q}$
0	X					
1	0					
1	1					

功能	S	R	$Q_{prev}$	Q
保持	0	0		$Q_{prev}$
清除	0	1	0	0
置位	1	0	1	1
非法	1	1		

# D Latch Internal Circuit



<i>CLK</i>	<i>D</i>	<i><math>\overline{D}</math></i>	<i>S</i>	<i>R</i>	<i>Q</i>	<i><math>\overline{Q}</math></i>
0	X	$\overline{X}$	0	0	$Q_{prev}$	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

功能	S	R	$Q_{prev}$	Q
保持	0	0		$Q_{prev}$
置0	0	1	0	0
置1	1	0	1	1
非法	1	1		

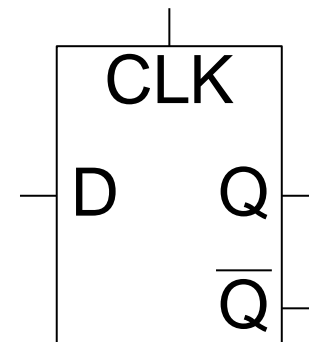
用D和CLK区分开了值与时间



# D Latch

- Two inputs:  $CLK$ ,  $D$ 
  - $CLK$ : controls *when* the output changes
  - $D$  (the data input): controls *what* the output changes to
- Function
  - When  $CLK = 1$ ,  
 $D$  passes through to  $Q$  (*transparent*)
  - When  $CLK = 0$ ,  
 $Q$  holds its previous value (*opaque*)
- Avoids invalid case when  
 $Q \neq \text{NOT } \bar{Q}$

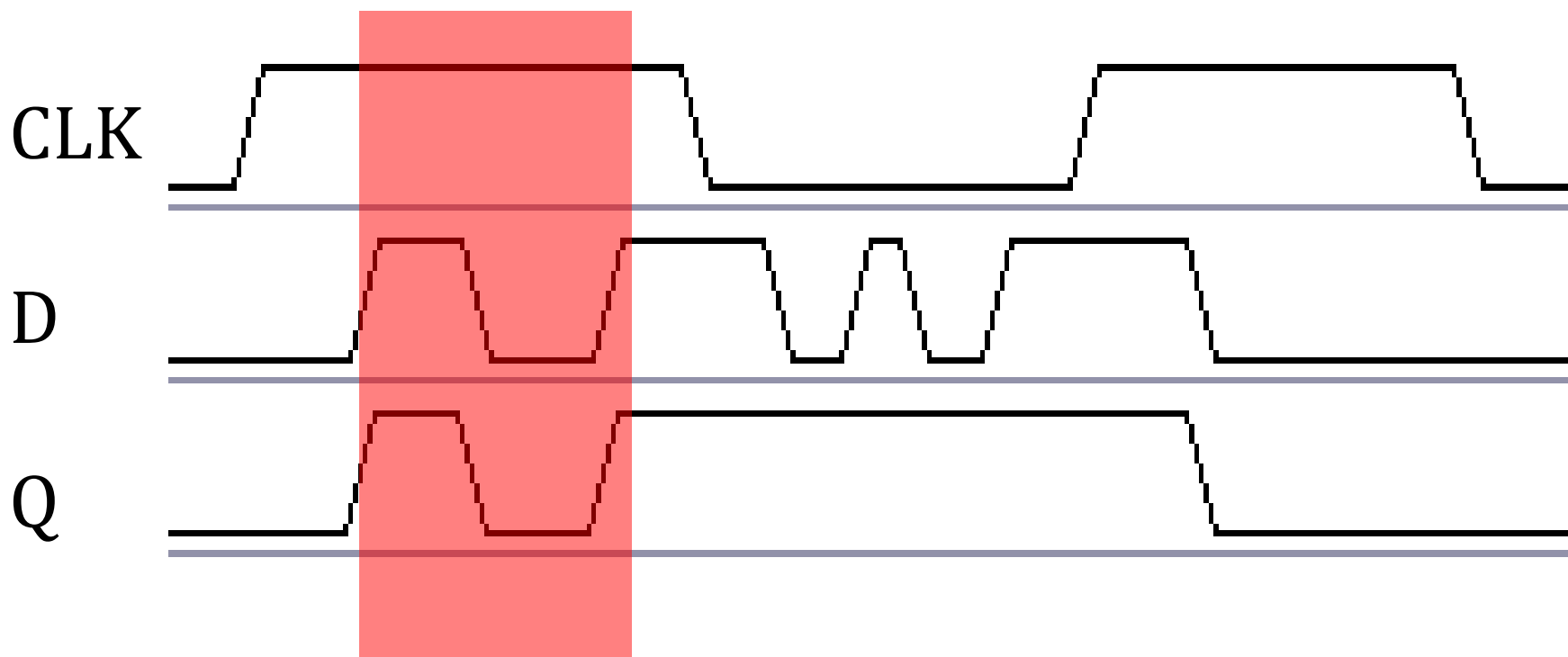
D Latch  
Symbol



# D Latch的局限

- 电平缓冲器

- 在一个时钟周期内，可以多次翻转
- 不符合定义时钟的设计初衷

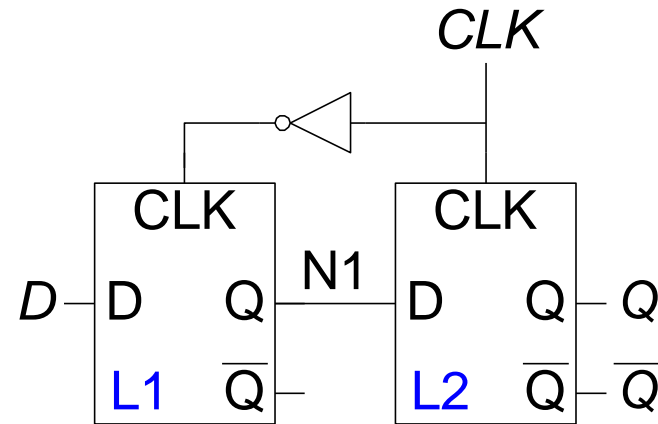


# D Flip-Flop Internal Circuit

- Two back-to-back latches (L1 and L2) controlled by complementary clocks

- When  $CLK = 0$

- L1 is transparent
- L2 is opaque
- $D$  passes through to N1



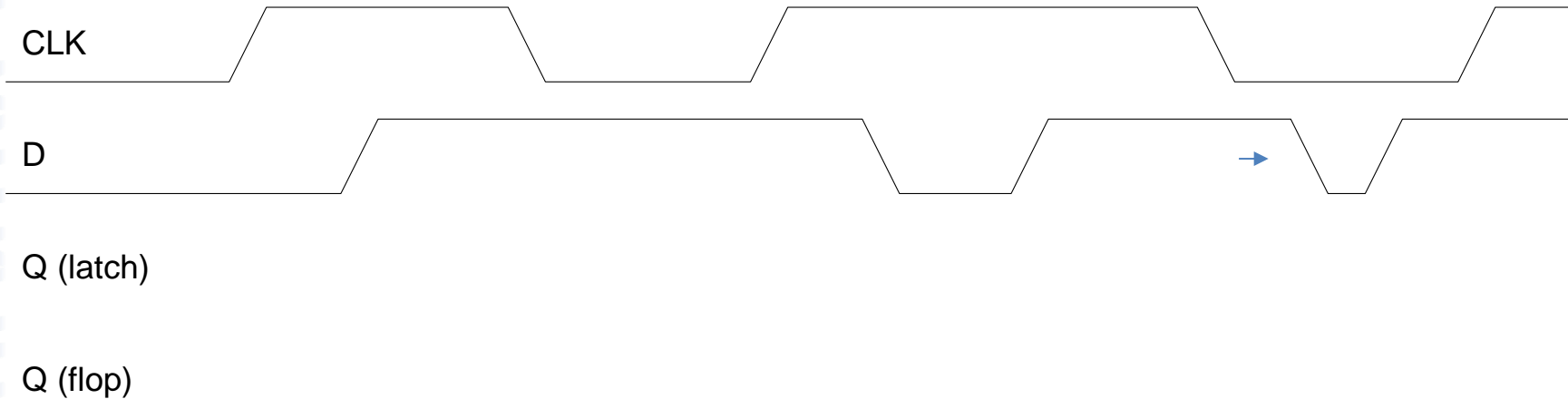
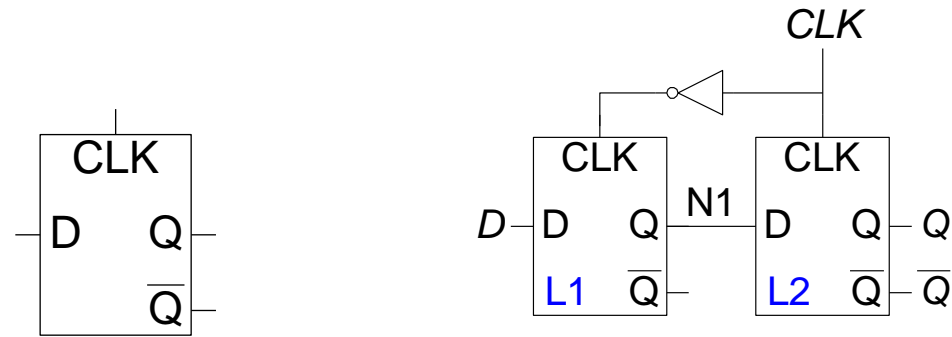
- When  $CLK = 1$

- L2 is transparent
- L1 is opaque
- N1 passes through to  $Q$

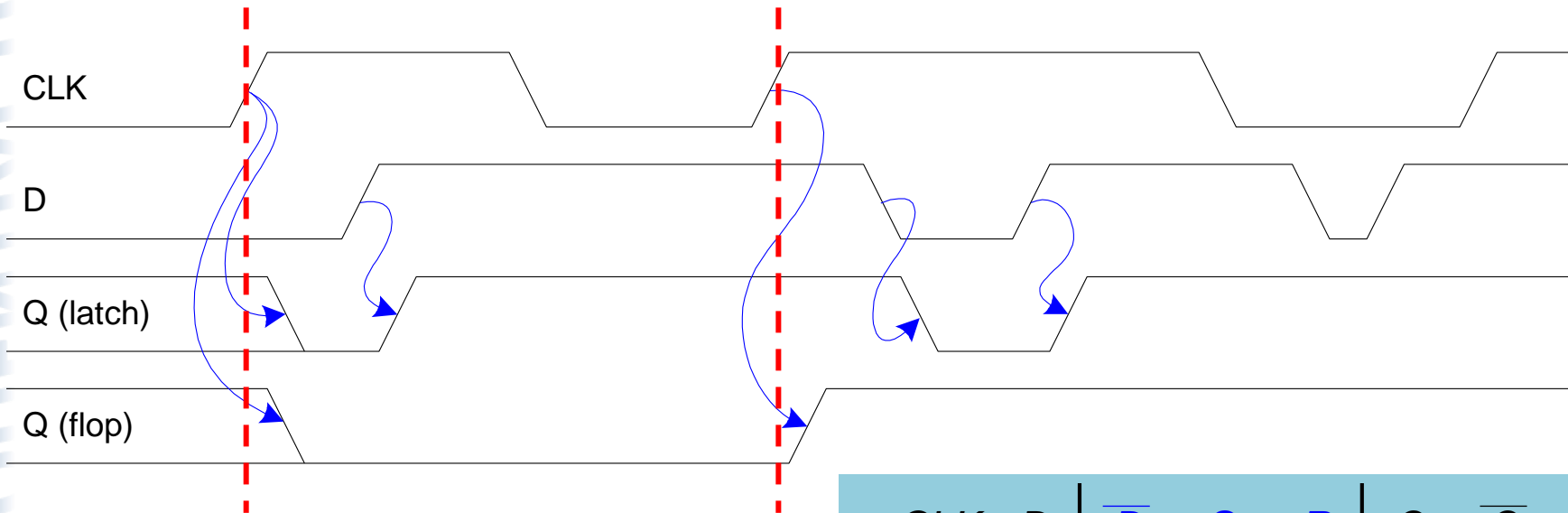
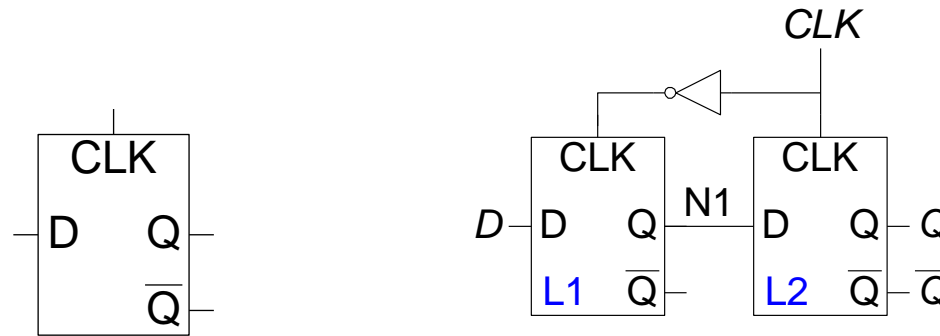
- Thus, on the edge of the clock (when  $CLK$  rises from 0 to 1)
  - $D$  passes through to  $Q$

$CLK$	$D$	$\overline{D}$	$S$	$R$	$Q$	$\overline{Q}$
0	X	$\overline{X}$	0	0	$Q_{prev}$	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

# D Latch vs. D Flip-Flop



# D Latch vs. D Flip-Flop

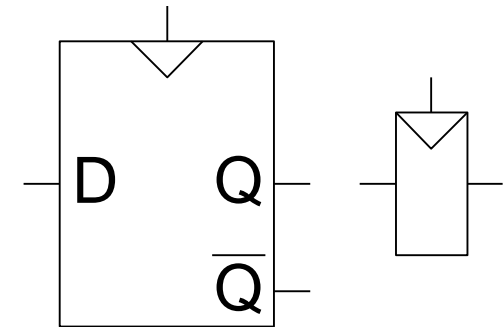


CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	$Q_{prev}$	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

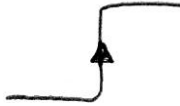
# D Flip-Flop

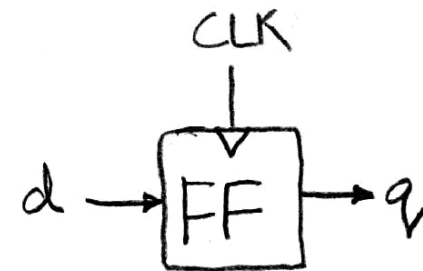
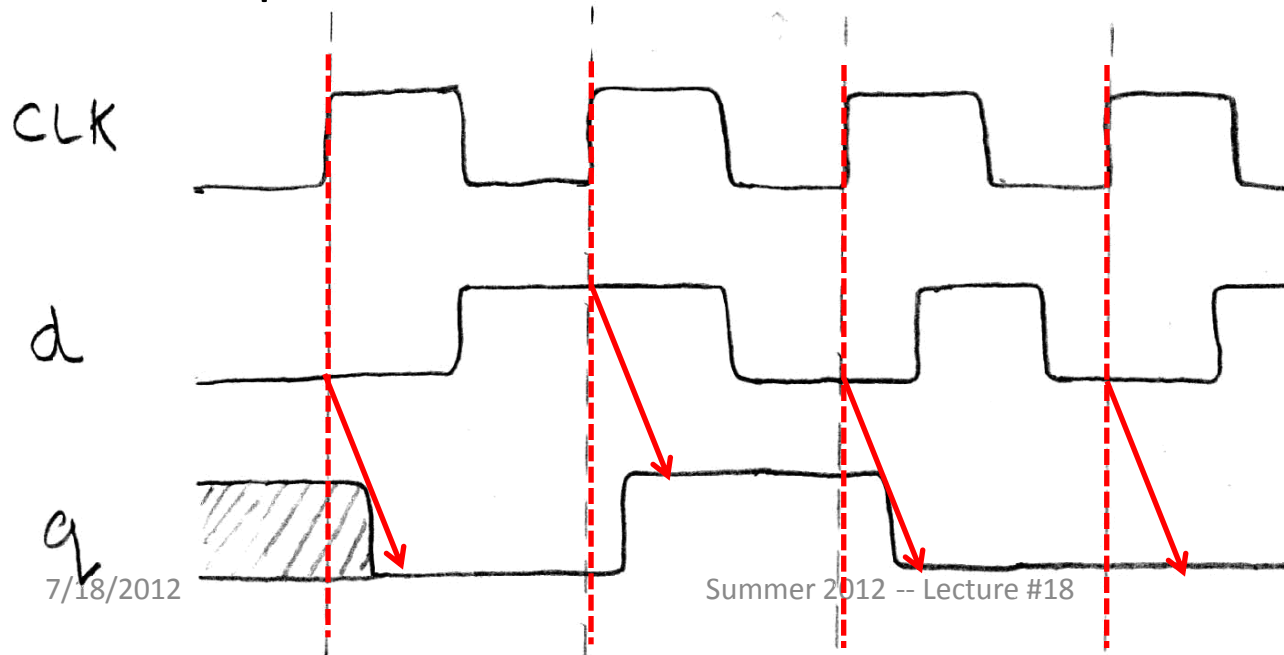
- **Inputs:**  $CLK$ ,  $D$
- **Function**
  - Samples  $D$  on rising edge of  $CLK$ 
    - When  $CLK$  rises from 0 to 1,  $D$  passes through to  $Q$
    - Otherwise,  $Q$  holds its previous value
  - $Q$  changes only on rising edge of  $CLK$
- Called *edge-triggered*
- Activated on the clock edge

D Flip-Flop Symbols



# Flip-Flop Timing Behavior (1/2)

- Edge-triggered D-type flip-flop
  - This one is “positive edge-triggered” 
- “On the rising edge of the clock, input d is sampled and transferred to the output. At other times, the input d is ignored and the previously sampled value is retained.”
- Example waveforms:



# Flip-Flop Timing Terminology (1/2)


- Camera Analogy: Taking a photo
  - *Setup time*: don't move since about to take picture (open camera shutter)
  - *Hold time*: need to hold still after shutter opens until camera shutter closes
  - *Time to data*: time from open shutter until image appears on the output (viewfinder)

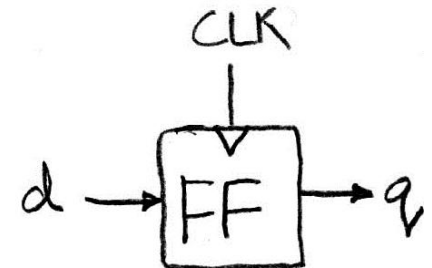
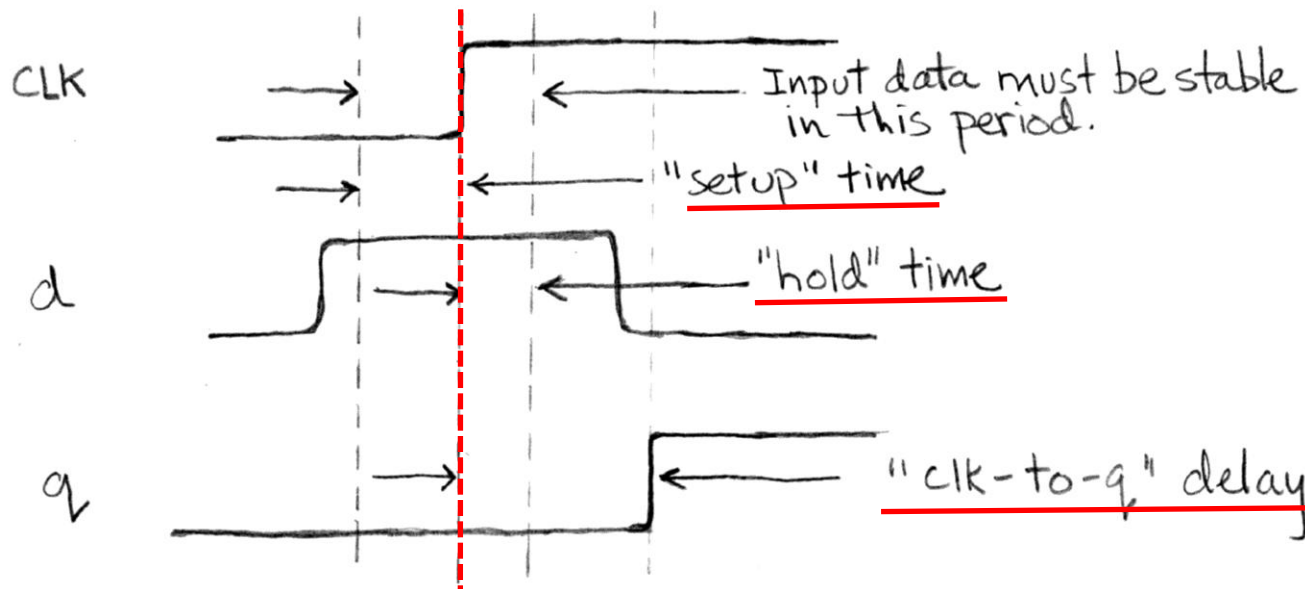


# Flip-Flop Timing Terminology (2/2)

- Now applied to hardware:
  - *Setup Time*: how long the input must be stable *before* the CLK trigger for proper input read
  - *Hold Time*: how long the input must be stable *after* the CLK trigger for proper input read
  - *“CLK-to-Q” Delay*: how long it takes the output to change, measured from the CLK trigger

# Flip-Flop Timing Behavior (2/2)

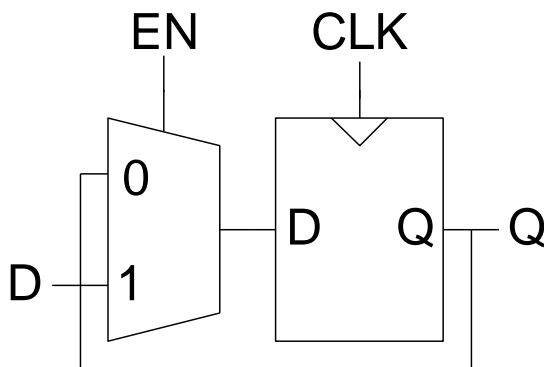
- Edge-triggered d-type flip-flop
  - This one is “positive edge-triggered” 
- “On the rising edge of the clock, input d is sampled and transferred to the output. At other times, the input d is ignored and the previously sampled value is retained.”



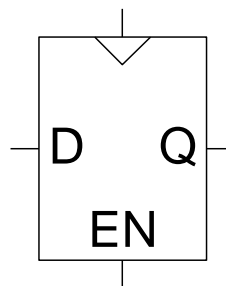
# Enabled Flip-Flops

- **Inputs:**  $CLK$ ,  $D$ ,  $EN$ 
  - The enable input ( $EN$ ) controls when new data ( $D$ ) is stored
- **Function**
  - $EN = 1$ :  $D$  passes through to  $Q$  on the clock edge
  - $EN = 0$ : the flip-flop retains its previous state

Internal  
Circuit



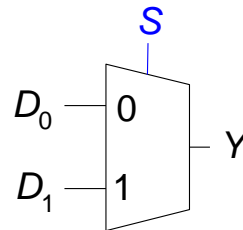
Symbol



# Multiplexer (Mux)

- Selects between one of  $N$  inputs to connect to output
- $\log_2 N$ -bit select input – control input
- Example:

## 2:1 Mux



$S$	$D_1$	$D_0$	$Y$	$S$	$Y$
0	0	0	0	0	$D_0$
0	0	1	1	1	$D_1$
0	1	0	0		
0	1	1	1		
1	0	0	0		
1	0	1	0		
1	1	0	1		
1	1	1	1		

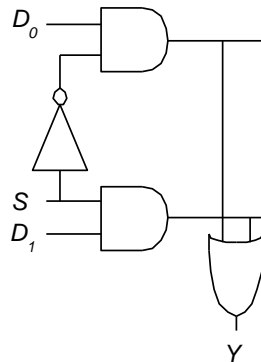
# Multiplexer Implementations

- **Logic gates**

- Sum-of-products form

Y S	$D_0 D_1$		00	01	11	10
	0	1	0	0	1	1
1	0	1	0	1	1	0

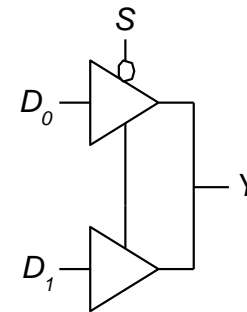
$$Y = D_0 \bar{S} + D_1 S$$



S	$D_1$	$D_0$	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

- **Tristates**

- For an N-input mux, use N tristates
- Turn on exactly one to select the appropriate input

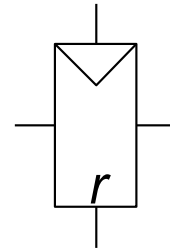
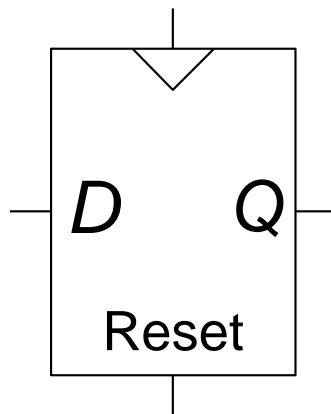


S	Y
0	$D_0$
1	$D_1$

# Resettable Flip-Flops

- **Inputs:**  $CLK$ ,  $D$ ,  $Reset$
- **Function:**
  - $Reset = 1$ :  $Q$  is forced to 0
  - $Reset = 0$ : flip-flop behaves as ordinary D flip-flop

## Symbols

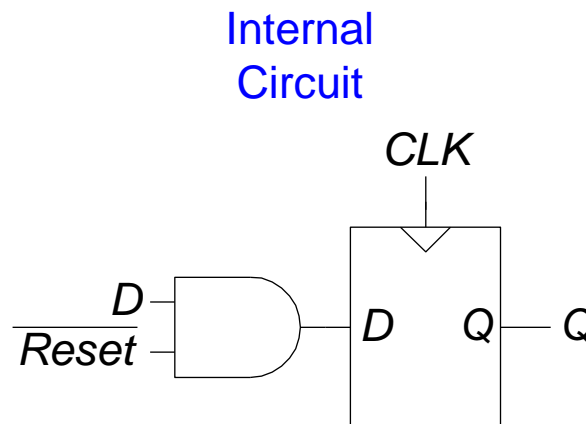


# Resettable Flip-Flops

- Two types:
  - **Synchronous:** resets at the clock edge only
  - **Asynchronous:** resets immediately when  $Reset = 1$
- Asynchronously resettable flip-flop requires changing the internal circuitry of the flip-flop
- Synchronously resettable flip-flop?

# Resettable Flip-Flops

- Two types:
  - **Synchronous:** resets at the clock edge only
  - **Asynchronous:** resets immediately when  $Reset = 1$
- Asynchronously resettable flip-flop requires changing the internal circuitry of the flip-flop
- Synchronously resettable flip-flop





# 作业

- 用组合逻辑设计一个可以完成循环左移功能的8位移位器
  - $D[7:0]$ ,  $Q[7:0]$ : 分别是移位器的8位输入和8个输出
  - $S[2:0]$ : 循环移位控制
    - ◆  $S[2:0]=000b$ , 无循环左移
    - ◆  $S[2:0]=001b$ , 循环左移1位
    - ◆ 依次类推
  - WORD:
    - ◆ 给出 $Q_7 \sim Q_0$ 的逻辑表达式
  - Logicsim:
    - ◆ 测试验证

# 作业

- 学习在Logicsim中输入真值表，自动综合出组合逻辑。电路功能如下：
  - 输入S[2:0]与Q[2:0]对应关系如下
    - ◆ 000: 010
    - ◆ 001: 101
    - ◆ 010: 110
    - ◆ 011: 111
    - ◆ 100: 000
    - ◆ 101: 001
    - ◆ 110: 010
    - ◆ 111: 011
  - 阅读User's Guide中的Combinational analysis，然后在logicsim中输入真值表，自动产生电路

# 作业

《数字设计》	Logicsim	WORD
2.24	✓	✓
3.1~3.3		✓
3.5	✓	✓

# 作业

## ■ 《数字设计与计算机体系结构》

《数字设计》	Logicsim	WORD
3.6	✓	1、给出真值表 2、简要说明工作原理
3.9	✓	
3.10	✓	

